Self-supervised learning and computer vision

Written: 13 Jan 2020 by *Jeremy Howard*

> *Update: Jan 20th, 2020*: Thanks to Yann LeCun for suggesting two papers from Facebook AI, Self-Supervised Learning of Pretext-Invariant Representations and Momentum Contrast for Unsupervised Visual Representation Learning. I've added a section "consistency loss" that discusses the approach used in these works (and similar ideas). Thanks for Phillip Isola for finding early uses of the term "self-supervised learning", which I've added to the post.

Introduction to self-supervised learning

Wherever possible, you should aim to start your neural network training with a pre-trained model, and fine tune it. You really don't want to be starting with random weights, because that's means that you're starting with a model that doesn't know how to do anything at all! With pretraining, you can use 1000x less data than starting from scratch.

So, what do you do if there are no pre-trained models in your domain? For instance, there are very few pre-trained models in the field of medical imaging. One interesting recent paper, Transfusion: Understanding Transfer Learning for Medical Imaging has looked at this question and identified that using even a few early layers from a pretrained ImageNet model can improve both the speed of training, and final accuracy, of medical imaging models. Therefore, you should use a general-purpose pre-trained model, even if it is not in the domain of the problem that you're working in.

However, as this paper notes, the amount of improvement from an ImageNet pretrained model when applied to medical imaging is not that great. We would like something which works better but doesn't will need a huge amount of data. The secret is "*self-supervised learning*". This is where we train a model using labels that are naturally part of the input data, rather than requiring separate external labels.

This idea has a long history, discussed back in 1989 by Jürgen Schmidhuber in his (way ahead of its time!) 1989 paper Making the World Differentiable. By 1994, the term was also being used to cover a related approach, which is using one modality as labels for another, such as the paper Learning Classification with Unlabeled Data, which uses audio data as labels, and video data as predictors. The paper gives the example:

> Hearing "mooing" and seeing cows tend to occur together

Self-supervised learning is the secret to ULMFiT, a natural language processing training approach that dramatically improves the state-of-the-art in this important field. In ULMFiT we start by pretraining a "*language model*" — that is, a model that learns to predict the next word of a sentence. We are not necessarily interested in the language model itself, but it turns out that the model which can complete this task must learn about the nature of language and even a bit about the world in the process of its training. When we'd then take this pretrained language model, and fine tune it for another task, such as sentiment analysis, it turns out that we can very quickly get state-of-the-art results with very little data. For more information about how this works, have a look at this introduction to ULMFiT and language model pretraining.

Self-supervised learning in computer vision

In self-supervised learning the task that we use for pretraining is known as the "*pretext task*". The tasks that we then use for fine tuning are known as the "*downstream tasks*". Even although self-supervised learning is nearly universally used in natural language processing nowadays, it is used much less in computer vision models than we might expect, given how well it works. Perhaps this is because

ImageNet pretraining has been so widely successful, so folks in communities such as medical imaging may be less familiar with the need for self-supervised learning. In the rest of this post I will endeavor to provide a brief introduction to the use of self-supervised learning in computer vision, in the hope that this might help more people take advantage of this very useful technique.

The most important question that needs to be answered in order to use self-supervised learning in computer vision is: "what pretext task should you use?" It turns out that there are many you can choose from. Here is a list of a few, and papers describing them, along with an image from a paper in each section showing the approach.

Colorization

(paper 1, paper 2, paper 3)

Placing image patches in the right place

(paper 1, paper 2)

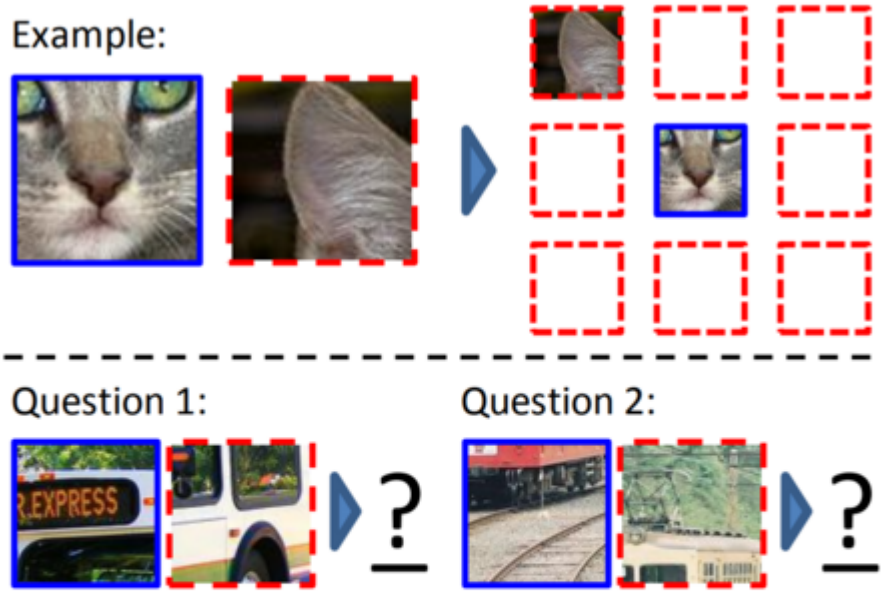
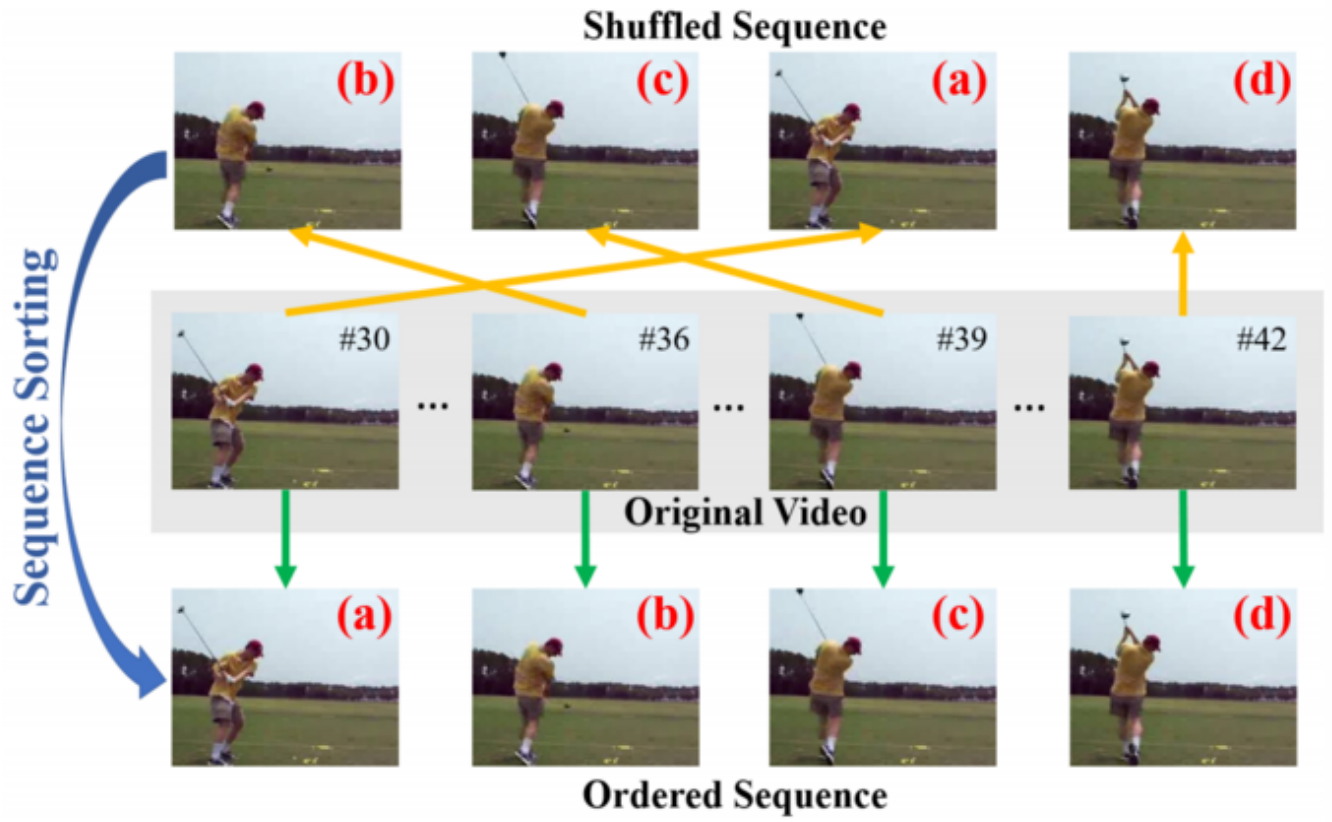


Figure 1. Our task for learning patch representations involves randomly sampling a patch (blue) and then one of eight possible neighbors (red). Can you guess the spatial configuration for the two pairs of patches? Note that the task is much easier once you have recognized the object!

Answer key: Q1: Bottom right Q2: Top center

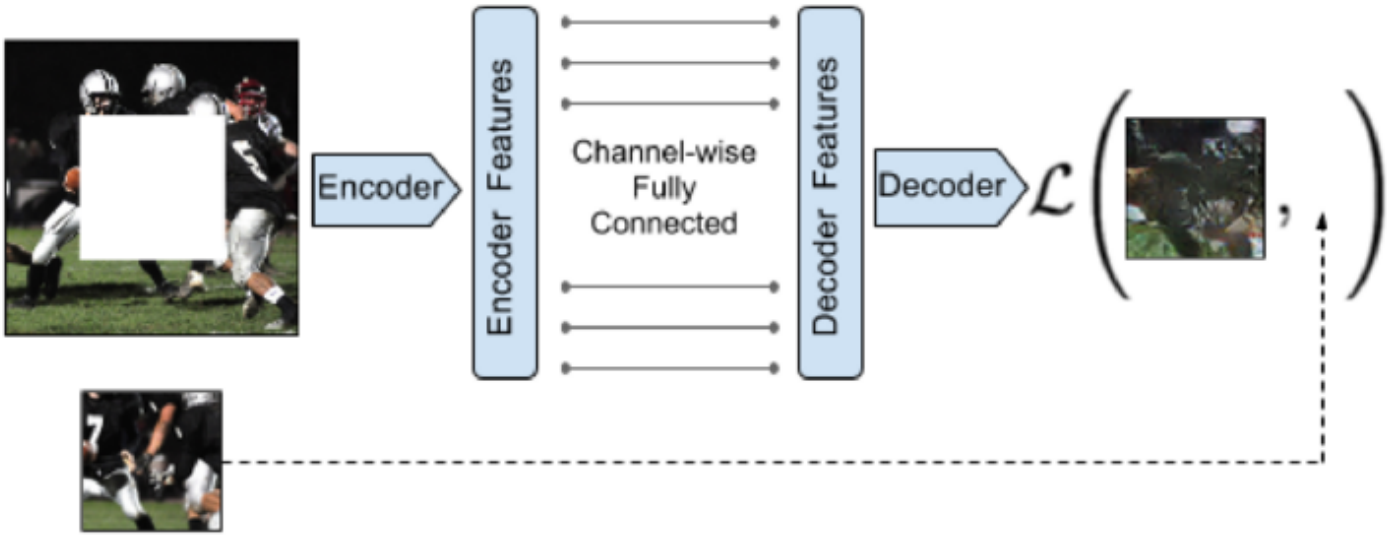
Placing frames in the right order

([paper 1](), [paper 2]())



Inpainting

(paper)



Classify corrupted images

(paper)

In this example, the green images are not corrupted, and the red images are corrupted. Note that an overly simple corruption scheme may result in a task that's too easy, and doesn't result in useful features. The paper above uses a clever approach that corrupts an autoencoder's features, and then tries to reconstruct them, to make it a challenging task.

Choosing a pretext task

The tasks that you choose needs to be something that, if solved, would require an understanding of your data which would also be needed to solve your downstream task. For instance, practitioners often used as a pretext task something called an "autoencoder". This is a model which can take an input image, converted into a greatly reduced form (using a bottleneck layer), and then convert it back into something as close as possible to the original image. It is effectively using compression as a pretext task. However, solving this task requires not just regenerating the original image content, but also regenerating any noise in the original image. Therefore, if your downstream task is something where you want to generate higher quality images, then this would be a poor choice of pretext task.

You should also ensure that the pretext task is something that a human could do. For instance, you might use as a pretext task the problem of generating a future frame of a video. But if the frame you try to generate is too far in the future then it may be part of a completely different scene, such that no model could hope to automatically generate it.

Fine tuning for your downstream tasks

Once you have pretrained your model with a pretext task, you can move on to fine tuning. At this point, you should treat this as a transfer learning problem, and therefore you should be careful not to hurt your pretrained weights. Use the things discussed in the ULMFiT paper to help you here, such as gradual unfreezing, discriminative learning rates, and one-cycle training. If you are using fastai2 then you can simply call the `fine_tune` method to have this all done for you.

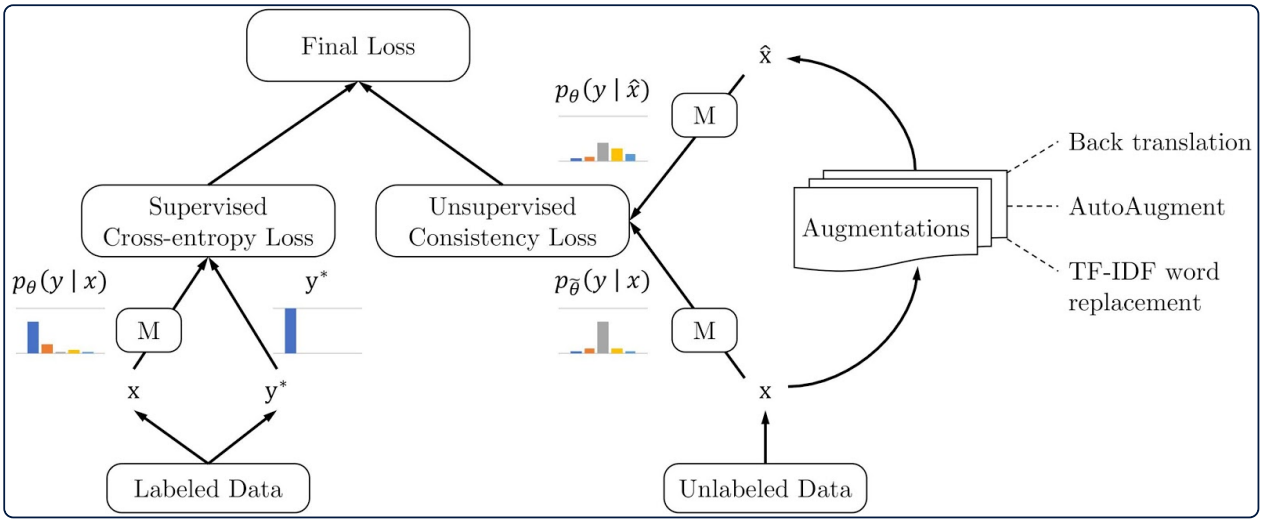
Overall, I would suggest not spending too much time creating the perfect pretext model, but just build whatever you can that is reasonably fast and easy. Then you can find out whether it is good enough for your downstream task. Often, it turns out that you don't need a particularly complex pretext task to get great results on your downstream task. Therefore, you could easily end up wasting time over engineering your pretext task.

Note also that you can do multiple rounds of self-supervised pretraining and regular pretraining. For instance, you could use one of the above approaches for initial pretraining, and then do segmentation for additional pretraining, and then finally train your downstream task. You could also do multiple tasks at once (multi-task learning) at either or both stages. But of course, do the simplest thing first, and then add complexity only if you determine you really need it!

Consistency loss

There's a very useful trick you can add on top of your self-supervised training, which is known as "consistency loss" in NLP, or "noise contrastive estimation" in computer vision. The basic idea is this: your pretext task is something that messes with your data, such as obscuring sections, rotating it, moving patches, or (in NLP) changing words or translating a sentence to a foreign language and back again. In each case, you'd hope that the original item and the "messed up" item give the same predictions in a pretext task, and create the same features in intermediate representations. And you'd also hope that the same item, when "messed up" in two different ways (e.g. an image rotated by two different amounts) should also have the same consistent representations.

Therefore, we add to the loss function something that penalizes getting different answers for different versions of the same data. Here's a pictorial representation, from Google's post [Advancing Semi-supervised Learning with Unsupervised Data Augmentation](#).



To say that this is "effective" would be a giant understatement... for instance, the approach discussed in the Google post above totally and absolutely smashed our previously state of the art approach to text classification with ULMFiT. They used 1000x less labeled data than we did!

Facebook AI has recently released two papers using this idea in a computer vision setting: [Self-Supervised Learning of Pretext-Invariant Representations](#) and [Momentum Contrast for Unsupervised Visual Representation Learning](#). Like the Google paper in NLP, these methods beat the previous state of the art approaches, and require less data.

It's likely that you can add a consistency loss to your model, for nearly any pretext task that you pick. Since it's so effective, I'd strongly recommend you give it a try!

Further reading

If you're interested in learning more about self-supervised learning in computer vision, have a look at these recent works:

- Self-supervised Visual Feature Learning with Deep Neural Networks: A Survey
- Revisiting Self-Supervised Visual Representation Learning
- Self-Supervised Representation Learning

This post is tagged: [`technical`] (click a tag for more posts in that category).

Related Posts

4 Principles for Responsible Government Use of Technology 21 Jan 2020

Your own blog with GitHub Pages and fast_template (4 part tutorial) 20 Jan 2020

Blogging with Jupyter Notebooks 20 Jan 2020